

LINKEDLIST

01 March 2025 13:23

What is LinkedList?

- LinkedList is a **doubly linked list implementation** in Java.

| | | |
|-----------------------|---------|-------------------|
| Previous node address | element | Next node address |
|-----------------------|---------|-------------------|

```
list.add("a");
list.add("b");
list.add("c");
```

| | | |
|------|---|-----|
| null | a | 102 |
| 101 | b | 103 |

| | | |
|-----|---|-----|
| 101 | b | 103 |
| 102 | | |

| | | |
|-----|---|------|
| 102 | c | null |
| 103 | | |

- It is part of **Java Collection Framework**.
- It implements both **List** and **Deque** interfaces, so it supports **list operations and queue operations** both.
- It is part of **java.util package**.
- It allows **duplicates** and **preserves insertion order**.
- It allows **null values**.

Why do we need LinkedList?

Problem with ArrayList

- In ArrayList, inserting or removing elements **in the middle** is **slow** because shifting is required.

Solution - LinkedList

- In LinkedList, **no shifting required**.
- It uses **nodes**, where each node stores:
 - **Data**
 - **Link to next node**
 - **Link to previous node**

Key Features of LinkedList

-  Doubly linked list structure
-  No shifting when adding/removing in middle
-  Maintains **insertion order**
-  Allows **duplicates**
-  Allows **null**
-  Supports **both list (index-based) and deque (queue methods)**



Common Methods of LinkedList

| Method | Description |
|---------------------|-------------------------|
| add(E e) | Add element at end |
| add(int index, E e) | Add at specific index |
| get(int index) | Get element at index |
| set(int index, E e) | Update element at index |
| remove(int index) | Remove element at index |
| remove(Object o) | Remove specific element |
| size() | Get number of elements |
| contains(Object o) | Check if element exists |
| isEmpty() | Check if list is empty |
| clear() | Remove all elements |
| toArray() | Convert to array |
| addFirst(E e) | Add element at start |
| addLast(E e) | Add element at end |
| removeFirst() | Remove first element |
| removeLast() | Remove last element |
| getFirst() | Get first element |

Creating LinkedList (Syntax)

```
import java.util.LinkedList;
LinkedList<String> list = new LinkedList<>();
```

Internal Working of LinkedList

- Internally uses **doubly linked nodes**.
- Each node has:
 - data (actual element)
 - next (link to next node)
 - prev (link to previous node)
- No shifting during insert/remove — only links are changed.

Advantages of LinkedList

- ✓ Fast insertion/removal from **beginning/middle/end**
- ✓ No need to resize
- ✓ Can work like **List** and **Queue/Deque** both
- ✓ Doubly linked for two-way traversal

◆ Disadvantages of LinkedList

- ✗ More memory (each node has 2 extra pointers — next, prev)
- ✗ Slower random access (no index array like ArrayList)

◆ When to Use LinkedList?

| Use Case | Recommended? |
|---|----------------------|
| ✓ Insert/remove frequently (especially in middle) | ✓ Yes |
| ✓ Need both List & Queue features | ✓ Yes |
| ✗ Fast random access needed | ✗ No (Use ArrayList) |
| ✗ Memory is a concern | ✗ No |

◆ Quick Difference: ArrayList vs LinkedList

| Feature | ArrayList | LinkedList |
|---------------------------|------------------------|---------------------------|
| Internal Structure | Array | Doubly Linked List |
| Memory | Compact | More (extra pointers) |
| Access Speed | Fast (direct index) | Slow (traverse from head) |
| Insert/Remove (Middle) | Slow (shifting needed) | Fast (just change links) |
| Insert/Remove (Start/End) | Slow | Fast |
| Part of | List interface | List + Deque interface |

◆ Real World Example for Explanation

| Scenario | Best Choice |
|-------------------------------|--|
| ✓ Passenger Queue | LinkedList (FIFO nature) |
| ✓ Students Roll Numbers List | ArrayList (Index access needed) |
| ✓ Undo History in Text Editor | LinkedList (Sequential backward/forward traversal) |

CODE :

```
package collection_framework;
import java.util.LinkedList;

public class LinkedListDemo {
    public static void main(String[] args) {
        LinkedList<String> list = new LinkedList<>();
```

```
list.add("Apple");
list.add("Banana");
list.add("Mango");

System.out.println("Original List: " + list);

list.addFirst("Orange");
list.addLast("Grapes");

System.out.println("After addFirst and addLast: " + list);

list.removeFirst();
list.removeLast();

System.out.println("After removeFirst and removeLast: " + list);

System.out.println("First element: " + list.getFirst());
System.out.println("Last element: " + list.getLast());

System.out.println("Contains Mango? " + list.contains("Mango"));
System.out.println("Index of Banana: " + list.indexOf("Banana"));

list.set(1, "Kiwi");
System.out.println("After set(1, Kiwi): " + list);

list.remove("Apple");
System.out.println("After removing Apple: " + list);

System.out.println("Size: " + list.size());

list.clear();
System.out.println("After clear, isEmpty? " + list.isEmpty());
}
```